

**WHAT IS CLAIMED IS:**

1   1. A computer-implemented method for processing software  
2   code, said method comprising:

3       receiving, at a second processor, a code processing  
4       request requested by a first processor, wherein the  
5       first and second processors are heterogeneous  
6       processors within a computer system that share a  
7       common memory;

8       writing software code data corresponding to the  
9       request to a local memory corresponding to the second  
10      processor in response to the request; and  
11      processing the software code data by the second  
12      processor.

1   2. The method as described in claim 1 further comprising:  
2       prior to the receiving:

3           reading script code from the common memory;

4           writing the script code to a local memory  
5           corresponding to the first processor;

6           interpreting, at the first processor, the script  
7           code, the interpreting resulting in the software  
8           code; and

9           writing the software code to the second  
10          processor's local memory.

1   3. The method as described in claim 1 further comprising:  
2       writing data resulting from the executing to the  
3       common memory.

1 4. The method as described in claim 1 further comprising:

2 prior to the receiving:

3 running a first program, during the running of the  
4 first program, identifying a call to the software  
5 code; and

6 loading the software code into the common memory,  
7 wherein the processing of the software code is occurs  
8 simultaneously to the running of the first program and  
9 wherein the processing is completed prior to the call  
10 of the software code from the first program.

1 5. The method as described in claim 4 further comprising:

2 performing a multimedia effect resulting from the  
3 processing of the software code, the performance  
4 performed by the second processor.

1 6. The method as described in claim 4 further comprising:

2 receiving, at the first processor, executable  
3 instructions resulting from the processing performed  
4 by the second processor, wherein the executable  
5 instructions are adapted to perform a multimedia  
6 effect; and

7 performing the multimedia effect on the first  
8 processor by executing the received executable  
9 instructions.

1 7. The method as described in claim 1 further comprising:

2 loading, a the second processor, a virtual machine  
3 program into the second processor's local memory;

4       reading, from the common memory shared by the first  
5       and second processors, the software code data that  
6       includes virtual machine code adapted to be processed  
7       by the virtual machine program;  
8       processing the virtual machine code at the second  
9       processor using the virtual machine program, the  
10      processing resulting in executable instructions;  
11      writing the executable instructions to a memory  
12      location accessible by the first processor using a DMA  
13      operation; and  
14      executing, at the first processor, the executable  
15      instructions.

1   8.   The method as described in claim 7 wherein the memory  
2       location is selected from the group consisting of a  
3       local memory corresponding to the first processor, and  
4       the common memory.

1   9.   The method as described in claim 7 wherein the first  
2       and second processors are dislike processors with  
3       different instruction set architectures and wherein  
4       the executable instructions are adapted to be executed  
5       on the first processor and not the second processor.

1   10.   The method as described in claim 1 wherein the  
2       processing results in one or more program instructions  
3       adapted to be performed by the first processor, the  
4       method further comprising:  
5       writing the program instructions to the common memory;

6        notifying the first processor that the program  
7        instructions have been written; and  
8        executing the program instructions by the first  
9        processor.

1    11. An information handling system comprising:  
2        a plurality of heterogeneous processors;  
3        a common memory shared by the plurality of  
4        heterogeneous processors;  
5        a first processor selected from the plurality of  
6        processors that sends a code processing request to a  
7        second processor, the second processor also being  
8        selected from the plurality of processors;  
9        a local memory corresponding to the second processor;  
10      a DMA controller associated with the second processor,  
11      the DMA controller adapted to transfer data between  
12      the common memory and the second processor's local  
13      memory; and  
14      a processing tool for processing software code, the  
15      processing tool including software effective to:  
16            receive, at a second processor, the code  
17            processing request requested by the first  
18            processor;  
19            write software code data corresponding to the  
20            request to the second processor's local memory in  
21            response to the request; and  
22            process the software code data by the second  
23            processor.

1   12. The information handling system as described in claim  
2   11 further comprising software code effective to:  
3       prior to the reception of the request:  
4           read script code from the common memory;  
5           write the script code to a local memory  
6           corresponding to the first processor;  
7           interpret, at the first processor, the script  
8           code, the interpreting resulting in the software  
9           code; and  
10          write the software code to the second processor's  
11        local memory.

1   13. The information handling system as described in claim  
2   11 further comprising software code effective to:  
3       write data resulting from the executing to the common  
4       memory.

1   14. The information handling system as described in claim  
2   11 further comprising software code effective to:  
3       prior to the reception of the request:  
4           run a first program, during the running of the  
5           first program, identify a call to the software  
6           code; and  
7           load the software code into the common memory,  
8           wherein the processing of the software code is  
9           occurs simultaneously to the running of the first  
10          program and wherein the processing of the

11               software code is completed prior to the call of  
12               the software code from the first program.

1   15. The information handling system as described in claim  
2   14 further comprising software code effective to:  
3       perform a multimedia effect resulting from the  
4       processing of the software code, the performance  
5       performed by the second processor.

1   16. The information handling system as described in claim  
2   14 further comprising software code effective to:  
3       receive, at the first processor, executable  
4       instructions resulting from the processing performed  
5       by the second processor, wherein the executable  
6       instructions are adapted to perform a multimedia  
7       effect; and  
8       perform the multimedia effect on the first processor  
9       by executing the received executable instructions.

1   17. The information handling system as described in claim  
2   11 further comprising software code effective to:  
3       load, at the second processor, a virtual machine  
4       program into the second processor's local memory;  
5       read, from the common memory shared by the first and  
6       second processors, the software code data that  
7       includes virtual machine code adapted to be processed  
8       by the virtual machine program;  
9       process the virtual machine code at the second  
10      processor using the virtual machine program, the  
11      processing resulting in executable instructions;

12        write, using a DMA operation, the executable  
13        instructions to a memory location accessible by the  
14        first processor; and  
  
15        execute, at the first processor, the executable  
16        instructions.

1        18. The information handling system as described in claim  
2        17 wherein the memory location is selected from the  
3        group consisting of a local memory corresponding to  
4        the first processor, and the common memory.

1        19. The information handling system as described in claim  
2        17 wherein the first and second processors are dislike  
3        processors with different instruction set  
4        architectures and wherein the executable instructions  
5        are adapted to be executed on the first processor and  
6        not the second processor.

1        20. The information handling system as described in claim  
2        11 wherein the process results in one or more program  
3        instructions adapted to be performed by the first  
4        processor, the information handling system further  
5        comprising software code effective to:  
  
6        write the program instructions to the common memory;  
  
7        notify the first processor that the program  
8        instructions have been written; and  
  
9        execute the program instructions by the first  
10      processor.

11 21. A computer program product stored on a computer  
12 operable media for processing software code, said  
13 computer program product comprising:  
  
14 means for receiving, at a second processor, a code  
15 processing request requested by a first processor,  
16 wherein the first and second processors are  
17 heterogeneous processors within a computer system that  
18 share a common memory;  
  
19 means for writing software code data corresponding to  
20 the request to a local memory corresponding to the  
21 second processor in response to the request; and  
  
22 means for processing the software code data by the  
23 second processor.

1 22. The computer program product as described in claim 21  
2 further comprising:

3 prior to the means for receiving:

4 means for reading script code from the common  
5 memory;

6 means for writing the script code to a local  
7 memory corresponding to the first processor;

8 means for interpreting, at the first processor,  
9 the script code, the interpreting resulting in  
10 the software code; and

11 means for writing the software code to the second  
12 processor's local memory.

1 23. The computer program product as described in claim 21  
2 further comprising:

3       means for writing data resulting from the executing to  
4       the common memory.

1   24. The computer program product as described in claim 21  
2       further comprising:

3       prior to the means for receiving:

4           means for running a first program, during the  
5           running of the first program, identifying a call  
6           to the software code; and

7           means for loading the software code into the  
8           common memory, wherein the processing of the  
9           software code is occurs simultaneously to the  
10          running of the first program and wherein the  
11          processing is completed prior to the call of the  
12          software code from the first program.

1   25. The computer program product as described in claim 24  
2       further comprising:

3       means for performing a multimedia effect resulting  
4       from the processing of the software code, the  
5       performance performed by the second processor.

1   26. The computer program product as described in claim 24  
2       further comprising:

3       means for receiving, at the first processor,  
4       executable instructions resulting from the processing  
5       performed by the second processor, wherein the  
6       executable instructions are adapted to perform a  
7       multimedia effect; and

8       means for performing the multimedia effect on the  
9       first processor by executing the received executable  
10      instructions.

1    27. The computer program product as described in claim 21  
2       further comprising:

3       means for loading, a the second processor, a virtual  
4       machine program into the second processor's local  
5       memory;

6       means for reading, from the common memory shared by  
7       the first and second processors, the software code  
8       data that includes virtual machine code adapted to be  
9       processed by the virtual machine program;

10      means for processing the virtual machine code at the  
11       second processor using the virtual machine program,  
12       the processing resulting in executable instructions;

13      means for writing the executable instructions to a  
14       memory location accessible by the first processor  
15       using a DMA operation; and

16      means for executing, at the first processor, the  
17       executable instructions.

1    28. The computer program product as described in claim 27  
2       wherein the memory location is selected from the group  
3       consisting of a local memory corresponding to the  
4       first processor, and the common memory.

1    29. The computer program product as described in claim 27  
2       wherein the first and second processors are dislike  
3       processors with different instruction set

4       architectures and wherein the executable instructions  
5       are adapted to be executed on the first processor and  
6       not the second processor.

1     30. The computer program product as described in claim 21  
2       wherein the means for processing results in one or  
3       more program instructions adapted to be performed by  
4       the first processor, the computer program product  
5       further comprising:

6       means for writing the program instructions to the  
7       common memory;

8       means for notifying the first processor that the  
9       program instructions have been written; and

10      means for executing the program instructions by the  
11       first processor.